Pure scripting

Table of contents

PPfilter-linked script structure 3
Shared functions 4
Getting Colors and Vectors 4
Functions 4
Camera 4
Sun and Moon angles 5
Weather 5
World 5
Postprocessing 5
Exposure 6
Sunangle based function 7
Gettings values from preparsed INI-files (PPfilter, Video, Graphics) 9
Using Pure's integrated exposure calculation 10
Using Pure's integrated picture improvements 11
Spectrum Adaption 11
VAO Adaption 11
CSP Lambert Gamma 11
Parameter access functions 12
Writing 12
Reading 12
Setting values absolute or relative 12
Setting an offset 12
Pure script parameter table 13
Custom UI item 15
Custom UI item definition 15
Custom UI item usage 17
Custom UI Items settings 18
Save Script Settings 18
Load Script Settings 18
Reset Script Settings 18
Custom weather variables 19
Initialize a weather variable 19

Specifying values for a certain weather 19
List of weather types: 19
Getting the value back 19
Skydome set scripts 20
Set structure 21
Texture 21
Texture functions 22
Weather and its sections 23
Weather names 23
Section functions 24
Functions Overview 25
Set functions 25
Texture functions 26
Weather functions 26
Section functions 26
Encryption 27
Encrypt all textures 27
Encrypt only specific textures 27
Generate the encrypted key 27
Set the key in your skydome set 27

PPfilter-linked script structure

A script must contain 2 functions: **init_pure_script** and **update_pure_script**

```
-- define some variables
local sunset_sat_boost = 3

-- This is called once while AC starts, or the ppfilter changed
function init_pure_script()

    -- Put static code in here, which is called only once, like activating Pure's own
exposure calculation
    Pure.exposure.useCBE(true)

    -- E.g. the Sunlight saturation is set to 2.0 once and is still adjustable by the
user.
    pure.config.set("light.sun.saturation", 2, true)

    -- E.g. the Sunlight saturation is set to 2.0 once and is NOT adjustable by the user.
    pure.config.set("light.sun.saturation", 2)
end

-- This is called every frame
function update_pure_script(dt)

    -- Put dynamic code in here, which is called every frame.
    -- E.g. the Sunlight saturation is boosted with sunset.
    pure.config.set("light.sun.saturation", pure.mod.sun(sunset_sat_boost), true)
end
```

A script will be automatically loaded, if the script is placed in the /system/cfg/ppfilters/pure_scripts folder and it has the same name like the current ppfilter with the file extension ".lua".

Shared functions

Getting Colors and Vectors

Use Pure's Color and Vector function to get its colors and vectors. Those functions returning secure core variables. No matter what you do, they will be restored in the next frame.

```
Pure.light.getLightsource()
Pure.light.getAmbient()
Pure.light.getDistantAmbient()
Pure.light.getDirectionalAmbient()
Pure.light.getSun()
Pure.light.getMoon()
Pure.light.getNLP()
Pure.light.getFog()

Pure.camera.getDirection()
Pure.camera.getPosition()
```

```
function update_pure_script(dt)
	ac.setGodraysCustomColor(Pure.light.getLightsource():scale(0.125))

	local amb = Pure.light.getAmbient()
	local amb_lum = amb:getLuminance()

	local camPos = Pure.camera.getPosition()
	ac.debug("Camera position", camPos)
end
```

Functions

Camera

Get the cameras direction or position vector.

```
Pure.camera.getDirection()
Pure.camera.getPosition()
```

Get the current camera FOV

```
Pure.camera.getFOV()
```

Get the current angles of the camera

```
Pure.camera.getHeading()
Pure.camera.getElevation()
```

Camera faces the Sun

```
PURE__CAMfacesSUN()
```

This is a very experimental function. Don't expect very precise output. It returns 0..1.
1 if the camera faces the sun to 0 if it does not face it. FOV is also taken in account. I tried to integrate CBE, to react on objects, which covers the sun.

Sun and Moon angles

```
Pure_get_Sun_horizantol_angle()
Pure.stellar.getSunElevation()
Pure.stellar.getMoonHeading()
Pure.stellar.getMoonElevation()
```

Weather

Get Pure's weather modifiers

```
Pure.world.getOvercast()
Pure.world.getFog()
Pure.world.getBadness()
```

World

Get the cloud shadow amount. This will return a calculated value for 3d and 2d clouds of the camera position.

```
Pure.world.getCloudShadow()
```

Get the luminance of the ambient light

```
Pure.light.getAmbientLuminance()
```

Get the multiplier, which controls the daylight. This will always be 1 at night and has the value of "light.daylight_multiplier" at day.

```
Pure.light.getSunLampDifference()
```

Set and get Stars properties

```
Pure.stellar.getStarsBrightness()
Pure.stellar.setStarsBrightness(1.0)
Pure.stellar.getStarsSaturation()
Pure.stellar.setStarsSaturation(1.0)
Pure.stellar.getStarsExponent()
Pure.stellar.setStarsExponent(1.0)
```

Postprocessing

Set Pure's Postprocessing parameters.

The "Curve" is a combined function. It will gain brightness and the mapping factor. So you can call it also "Gamma".

But you can adjust the mapping factor as its own.

```
Pure.pp.setTonemappingCurve(0.0)
PURE__set_PP_Tonemapping_FactorMultiplier(1.0)
```

Exposure

Set and get Pure's exposure calculation parameters

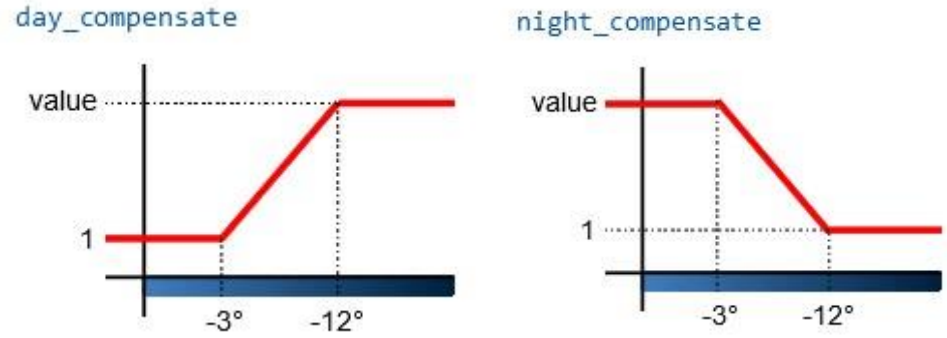
```
Pure.exposure.useCBE(true)
Pure.exposure.setCBEMix(1.0)
Pure.exposure.cbe.setLimits(0.05, 0.5)
Pure.exposure.cbe.setTarget(1.0)
Pure.exposure.cbe.setSensitivity(1.0)
Pure.exposure.cbe.setMultiplier(1.0)
Pure.exposure.cbe.setAdaptionSpeeds(1.0, 1.0)
Pure.exposure.cbe.setAdaptionSpeeds_YEBIS(1.0, 1.0)
Pure.exposure.setBypass(0.05)
Pure.exposure.cbe.getValue()
PURE__ExpCalc_get_YEBIS_calculated_exposure(false)
Pure.exposure.getValue()
```

Those functions are further described in chapter Using Pure's integrated exposure calculation

Sunangle based function

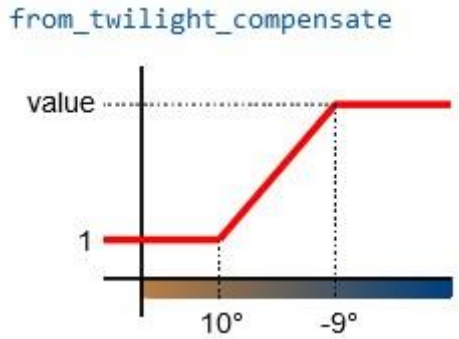
You can use several functions to manipulate your settings dynamically.
The compensate function returning the given value or 1 (its mainly used as a multiplicator)

Transition between v and 1, with sunangles between **-3° and -12°**



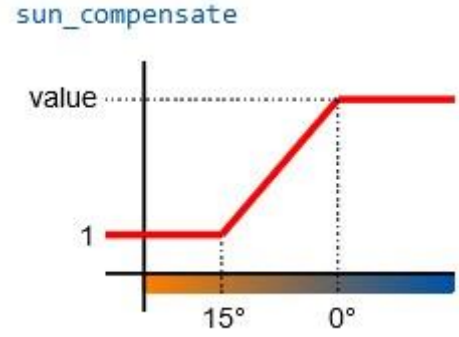
`pure.mod.day(v)` - returns 1 while day | v while night
`pure.mod.night(v)` - returns 1 while night | v while day

Transition between v and 1, with sunangles between **+10° and -9°**



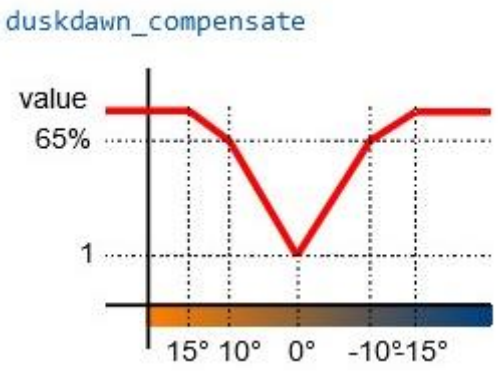
`pure.mod.twilight(v)` - returns 1 while day | v while night

Transition between v and 1, with sunangles between **+15° and 0°**



`pure.mod.sun(v)` - returns 1 with sun | v when sun is < 0°

Transition between v and 1, with sunangles between **+15° and -15°**



`pure.mod.duskdawn(v)` - returns 1 while dusk or dawn

Transition between v and 1, with sunangles between **+15° and -15°**

dawn_exclusive(v) - returns 1 while not in dawn
dusk_exclusive(v) - returns 1 while not in dusk

Transition between 2 values, with sunangles between **+90° and -0°**

`pure.mod.dayCurve(x, y, e)` - "Interpolate Day"

sin function of the sun angle, to interpolate between:

x (value when sunangle is 0°) and y (value when sun is 90°)

e is the exponent of the sin function a = sin(sunangle)^e
a is then the index of the linear interpolation of x and y
pure.mod.dayCurve returns math.lerp(x, y, a)

```
pure.mod.dayCurve(0, 1, 1.00) -- returns 0.347 at a sunangle of 20°
pure.mod.dayCurve(0, 1, 0.67) -- returns 0.487 at a sunangle of 20°
pure.mod.dayCurve(0, 1, 0.40) -- returns 0.653 at a sunangle of 20°
```

E.g. Using `pure.mod.day()` to fade from 1.1 to 1.3 with setting sun

"Boost PP Brightness in nighttimes"

1. Method:

```
ac.setPpBrightness( 0.1 + pure.mod.day(1.2) )
```

2. Method (using math.lerp):

```
ac.setPpBrightness( math.lerp( 1.3, 1.1, pure.mod.day(0) ) )
```

Gettings values from preparsed INI-files (PPfilter, Video, Graphics)

Get Values from the current used PPfilter

```
Pure.file.getPPfilterEntry("GLARE","THRESHOLD")
```

Get Values from the User's video.ini

```
Pure.file.getVideoEntry("CAMERA","MODE")
```

Get Values from the AC's system graphics.ini

```
Pure.file.getGraphicsEntry("DX11","SKYBOX_REFLECTION_GAIN")
```

Using Pure's integrated exposure calculation

Pure will calculate the right exposure from CSP's Cubemap Brightness Estimation. This works no matter if PostProcessing is used, Autoexposure is on or off, or even with PostProcessing deactivated.

Activate it in the init function. If you set it to 1.0, only Pure's exposure calculation is working. But you can also mix Pure's exposure and YEBIS exposure, if you set a value between 0 and 1. If you set it to 0, then only YEBIS exposure (static or Autoexposure) is working.

```
function init_pure_script()
	Pure.exposure.setCBEMix(1.0)
end
```

The Target modulates the CBE value before the calculation. So, you force more or less light. The sensitivity is an exponent within the calculation. A higher exponent, results in higher reaction to lower light.

```
Pure.exposure.cbe.setTarget(1.5)

Pure.exposure.cbe.setSensitivity(1.5)
```

Set a minimum and maximum value for the calculation's result

```
Pure.exposure.cbe.setLimits(0.05, 0.5)
```

Set the speeds to react to lower or brighter light

A higher value means faster a reaction. The default is 2.5. If you set a high value for bright light, like 10 and a low value for dark light, like 1, you can emulate a human eye.

```
Pure.exposure.cbe.setAdaptionSpeeds(lowlight, brightlight)
```

Set the speeds of YEBIS Autoexposure

Pure adds a smoothing of YEBIS Autoexposure as well. If no speeds are set for it, the CBE speeds (you set with the function above) are used.

```
Pure.exposure.cbe.setAdaptionSpeeds_YEBIS(lowlight, brightlight)
```

Modifying its resulting value

```
Pure.exposure.cbe.setMultiplier(1.25)
```

Using Pure's integrated picture improvements

Spectrum Adaption

Esp. for overcast situations the overall color temperature needs to be adapted, because only ambient light is then active. This is activated by default.
You can set your own mix, just set a number. If the first parameter is "true", the mix will be 1.0.

```
Pure.light.setSpectrumAdaption(false)
Pure.light.setSpectrumAdaption(0.75)
```

VAO Adaption

Esp. for overcast situations the VAO component can be boosted, to improve the lights dynamic.
This is deactivated by default.
You can set your own mix, just set a number. If the first parameter is "true", the mix will be 1.0.

```
Pure.light.setVAOAdaption(true)
Pure.light.setVAOAdaption(0.75)
```

CSP Lambert Gamma

This will improve the look with low sun angles. Before CSP 1.79, the sun lighting of materials was only calculated in linear space. That leads to a big loss of sunlight on horizontal materials (landscape) with low sun angles. To compensate this, a gain of sunlight is done in Pure. But this also gains the lighting all materials and lighting is too bright.

With CSP 1.79 sunlight on materials can be gained by using a gamma value. So lower lighting is higher and the former hack can be removed.

Use this function to restore the normal sunlight and gain lower lighting on materials:

```
Pure.light.setLambertGamma(1.7)
```

Set it to -1, to have Pure's sunlight hack and linear lighting

```
Pure.light.setLambertGamma(-1)
```

A gamma of 1.0 will avoid Pure's sunlight hack, but materials still have linear lighting

```
Pure.light.setLambertGamma(1.0)
```

Parameter access functions

Writing

```
pure.config.set(<KEY>, value, <relative>)
```

This function also returns the set values, if it could set…

Reading

```
pure.config.get(<KEY>)
```

Setting values absolute or relative

Setting a multiplier

```
pure.config.set("light.sun.saturation", 1.05, true)
```

This will **multiply** the initial value of sunlight's saturation **by** 1.05.

```
pure.config.set("light.sun.saturation", 1.05)
```

This will **set** the sunlight's saturation **to** 1.05.

Setting an offset

```
pure.config.set("light.sun.hue", -20, true)
```

This will **add** -20 to the offset of sunlight's hue.

```
pure.config.set("light.sun.hue", -20)
```

This will **set** the offset of sunlight's hue **to** -20.

Have a look in Pure's parameter table, to know, if a parameter is a multiplier or an offset:

Multipliers are marked as ✱

Offsets are marked as +

Pure script parameter table

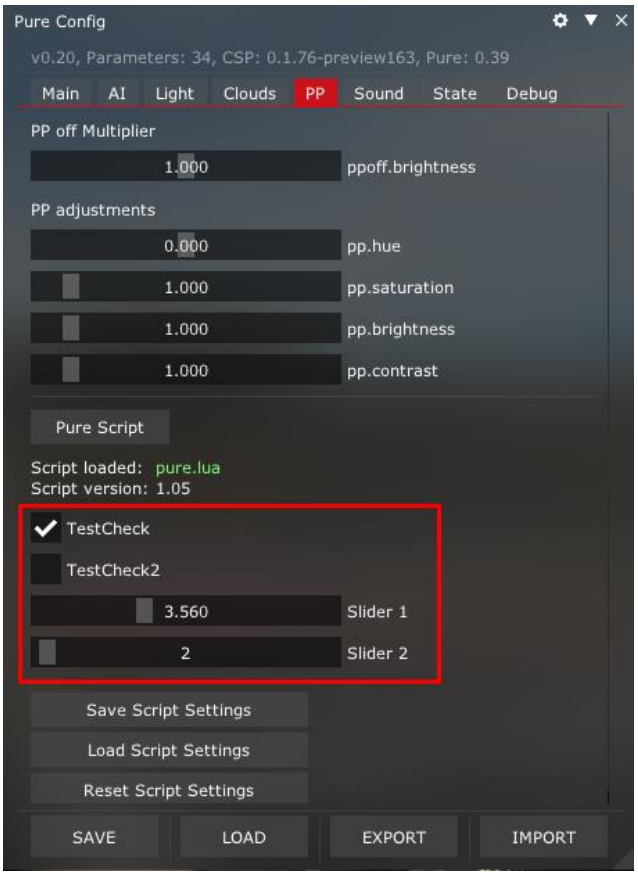
Some parameters need a reset of the weatherFX engine.
These parameters cannot be set in a Pure script!

<KEY>	Relative type	default	min	max	Reset
AI_headlights.ambient_light	*	10.00	0.00	20.00	
AI_headlights.CBE	*	0.60	0.00	5.00	
AI_headlights.fog	*	0.40	0.00	1.00	
light.daylight_multiplier	*	1.00	0.00	10.00	
light.sun.hue	+	0.00	-180	180	
light.sun.saturation	*	1.00	0.00	10.00	
light.sun.level	*	1.00	0.00	10.00	
light.ambient.hue	+	0.00	-180	180	
light.ambient.saturation	*	1.00	0.00	10.00	
light.ambient.level	*	1.00	0.00	10.00	
light.directional_ambient.hue	+	0.00	-180	180	
light.directional_ambient.saturation	*	1.00	0.00	10.00	
light.directional_ambient.level	*	1.00	0.00	10.00	
light.sky.hue	+	0.00	-180	180	
light.sky.saturation	*	1.00	0.00	10.00	
light.sky.level	*	1.00	0.00	10.00	
csp_lights.bounce	*	1.00	0.00	10.00	
csp_lights.emissive	*	1.00	0.00	10.00	
reflections.saturation	*	1.00	0.00	10.00	
reflections.level	*	1.00	0.00	10.00	
reflections.emissive_boost	*	8.00	0.00	50.00	
vao.amount	*	1.00	0.00	2.00	
vao.track_exponent	*	0.50	0.00	2.00	
vao.dynamic_exponent	*	0.33	0.00	2.00	
ui.white_reference_point	*	1.00	0.00	10.00	
nlp.level	*	1.00	0.00	10.00	
nlp.density	*	1.00	0.00	10.00	
nlp.lowest_ambient	*	1.00	0.00	10.00	
clouds_render		0	0	1	Reset
clouds.distance		1.00	0.50	3.00	Reset
clouds.quality		0.85	0.00	4.00	Reset
clouds.advanced_lighting		true			Reset
clouds.position_calculation_mode		1	0	2	Reset
clouds.position_calculation_limiter		500	-1	500	Reset
clouds.color_calculation_limiter		100	-1	500	Reset
clouds2D.set		“default”			Reset
clouds2D.quality		3	1	4	Reset
clouds2D.crossfade_time		5.00	1.00	20.00	
clouds2D.advanced_shadows		true			Reset
clouds2D.advanced_shadows_sun_cover		true			Reset
clouds2D.advanced_shadows_speed		1.00	0.10	10.00	
clouds2D.preload		false			Reset
clouds2D.unload		true			Reset
clouds2D.brightness		1.00	0.00	10.00	
clouds2D.contrast		1.00	0.00	10.00	
ppoff.brightness=1	*	1.00	0.00	2.00	
pp.hue	+	0.00	-180	180	
pp.saturation	*	1.00	0.00	10.00	
pp.brightness	*	1.00	0.00	10.00	
pp.contrast	*	1.00	0.00	10.00	
sound.wind_volume_interior	*	0.25	0.00	1.00	
sound.wind_volume_exterior	*	1.00	0.00	1.00	
sound.rain_volume_interior	*	0.80	0.00	1.00	
sound.rain_volume_exterior	*	1.00	0.00	1.00	

debug.memory debug.graphics		false false			Reset Reset

Custom UI item

Use custom UI items in Pure Config app



Custom UI item definition

Define UI items in the init function:

```
function init_pure_script()
	pure.script.setVersion(1.05)
	pure.script.resetSettingsWithNewVersion()

	pure.script.ui.addText("Lorem ipsum dolor sit amet, consetetur sadipscing")
	pure.script.ui.addCheckbox("TestCheck", true)
	pure.script.ui.addCheckbox("TestCheck2", false, "Tooltip")
	pure.script.ui.addSliderFloat("Slider 1", 3.56, 0, 10, "Just a slider")
	pure.script.ui.addSliderInteger("Slider 2", 2, 1, 53, "a second slider")

	pure.script.ui.addSeparator()

	pure.script.ui.addStateFloat("Weight of 1 apple", 0)
	pure.script.ui.addStateInteger("Number of apples", 0)
	pure.script.ui.addStateInteger("Taste of an apple", "delicious")
end
```

Set a version number

```
pure.script.setVersion(v: float)
```

Delete the settings file with a new version of the script

```
pure.script.resetSettingsWithNewVersion()
```

Define a Checkbox

```
pure.script.ui.addCheckbox(name: string, checked: Boolean, tooltip: string)
```

Define a Slider for float values

```
pure.script.ui.addSliderFloat(name: string, value: float, minimum: float, maximum: float,
tooltip: string)
```

Define a Slider for integer values

```
pure.script.ui.addSliderFloat(name: string, value: integer, minimum: integer, maximum:
integer, tooltip: string)
```

Define a Text

```
pure.script.ui.addText(text: string)
```

Define a Separator (just a thin line)

```
pure.script.ui.addSeparator()
```

Defining States (Values or Strings to monitor something in the Pure Config app)

Float values

```
pure.script.ui.addStateFloat(name: string, initial value: float)
```

Integer values

```
pure.script.ui.addStateInteger(name: string, initial value: integer)
```

Strings

```
pure.script.ui.addStateInteger(name: string, initial text: string)
```

Custom UI item usage

Retrieve the values of custom UI items

```
function update_pure_script(dt)
	if pure.script.ui.getValue("TestCheck2") then
		pure.config.set("light.sky.level", pure.script.ui.getValue("Slider 1"))
	else
		pure.config.set("light.sky.level", 1.00)
	end
end
```

```
pure.script.ui.getValue(name: string): Boolean | float | integer
```

The function retrieves the value of the item with the “name”. It retrieves a boolean value for Checkboxes, float values for Float-Slider and an integer value for Integer-Sliders.

Set State values

```
function update_pure_script(dt)
	pure.script.ui.setValue("Weight of 1 apple", apple_weight)
	pure.script.ui.setValue("Number of apples", n_apples)
	pure.script.ui.addStateInteger("Taste of an apple", apple_taste)
end
```

```
pure.script.ui.setValue(name: string, value: float | integer)
```

Use the same function to set float and integer values. If you define a state as Integer, the given value is rounded and presented as Integer.

```
pure.script.ui.addStateInteger(name: string, text: string)
```

You can also use String States to display your custom string formation:

Use string.format to easily build that string, e.g.:

```
pure.script.ui.setString("String Test", string.format("1 apple weighs %.2f kg and tastes
%s", apple_weight, apple_taste))
```

Custom UI Items settings

If custom UI items are provided for the users, it is possible to save the user`s individual settings

Save Script Settings

The current settings of the UI items are saved in the file:
<script_name>_ScriptSettings.ini
Folder location:
\assettocorsa\extension\config-ext\Pure

Load Script Settings

Load settings from \assettocorsa\extension\config-ext\Pure\<script_name>_ScriptSettings.ini

Reset Script Settings

Reset all items to their default values, given at their definition.

Custom weather variables

Initialize a weather variable

```
pure.script.weather.addVariable(name, default_value)
```

This adds a new weather variable to the script. The default value is established to all weather.

```
pure.script.weather.addVariable("contrast", 1.0)
```

Specifying values for a certain weather

```
pure.script.weather.setVariable(name, weather-type, value)
```

This function will specify a certain value for a specific weather type.

```
pure.script.weather.setVariable("contrast", “BrokenClouds”, 1.05)
```

List of weather types:

```
"LightThunderstorm", "Thunderstorm", "HeavyThunderstorm"
"LightDrizzle", "Drizzle", "HeavyDrizzle"
"LightRain", "Rain", "HeavyRain"
"LightSnow", "Snow", "HeavySnow"
"LightSleet", "Sleet", "HeavySleet"
"Clear"
"FewClouds", "ScatteredClouds", "BrokenClouds"
"OvercastClouds"
"Fog", "Mist", "Smoke", "Haze", "Sand", "Dust"
"Squalls", "Tornado", "Hurricane", "Windy"
"Hail"
"NoClouds"
```

Getting the value back

```
pure.script.weather.getVariable(name)
```

This will return a correct interpolated value in dynamic weather.

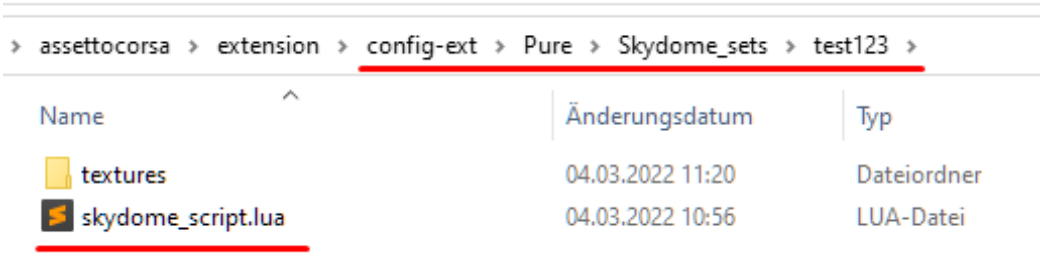
```
ac.setPpContrast( pure.script.weather.getVariable("contrast") )
```

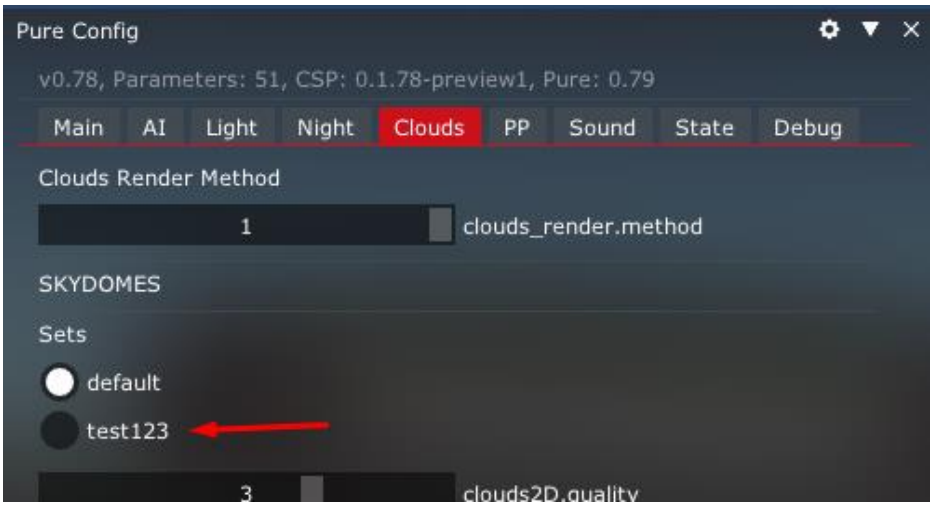
If you set 1.05 for “BrokenClouds” then it will give you a contrast of 1.05 only for broken clouds weather. Otherwise it is 1.0. While a weather change from or to “BrokenClouds”, it will interpolate this value, for a smooth fade.

Skydome set scripts

A skydome set will be shown as a set in Pure Config app, if:

1. there is a folder in **/extension/config-ext/Pure/Skydome_sets/**
2. this folder contains a script named "**skydome_script.lua**".





The set is initialized by calling the INIT function:

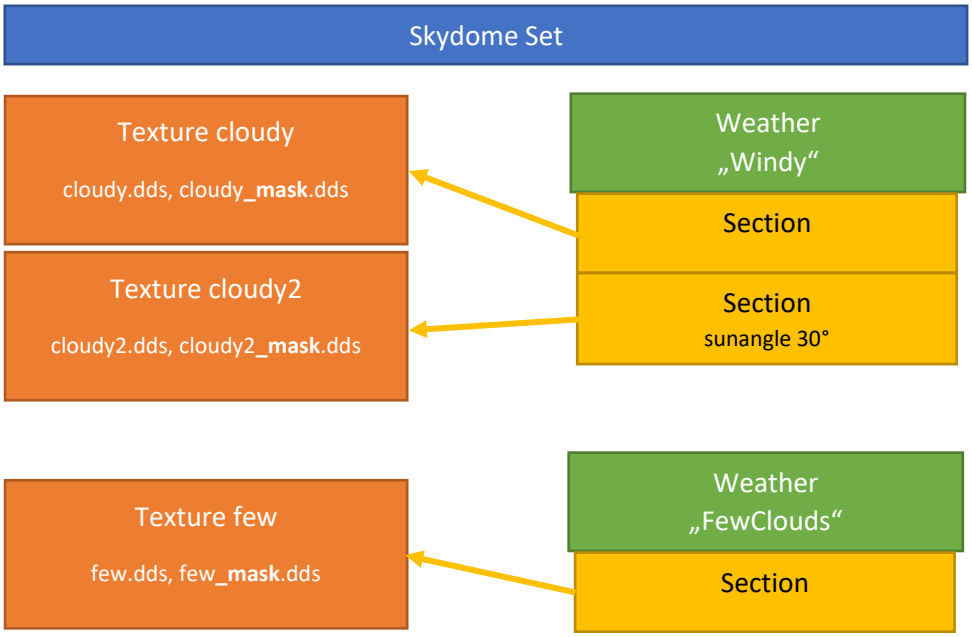
```
function init_skydome_set()

end
```

Within this init function, you must define all settings!

Set structure

A skydome set consists of textures, weather and sections.



Texture

The texture object contains the texture and an optional mask.
You must name those files like this:

- cloudy.dds

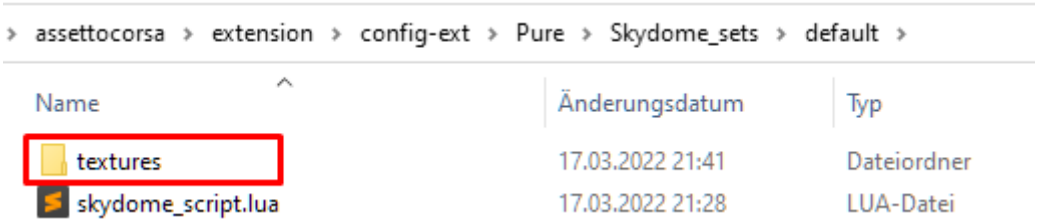
Additionally, you can create a mask file. This file will be used as an alpha mask. So black makes the texture transparent and white makes it opaque. You must name this file:

- cloudy**_mask**.dds

I recommend to export your textures to DDS BC7 format. This will speed up the loading times in AC tremendously.

To export DDS BC7 you need [NVIDIA's Texture Tools Exporter](#) !
If you use Photoshop, you can install its plugin.

Textures must be located in the /textures subfolder of the set's folder.



You can add a texture to the system with this command:

- PURE__2DCLOUDS_add_texture(name, file)

```
PURE__2DCLOUDS_add_texture("MediumClouds_Noon", "cloudy")
```

When adding the texture, the function PURE__2DCLOUDS_add_texture returns a texture object. You can the texture's function via that object.

```
texture = PURE__2DCLOUDS_add_texture("MediumClouds_Noon", "cloudy")
texture:setHorizonOffset(-0.014)
texture:setSunPosition(0.1)
texture:setSkyAdaption(0, 0.5, 0.5)
texture:setBrightness(1.0)
```

Texture functions

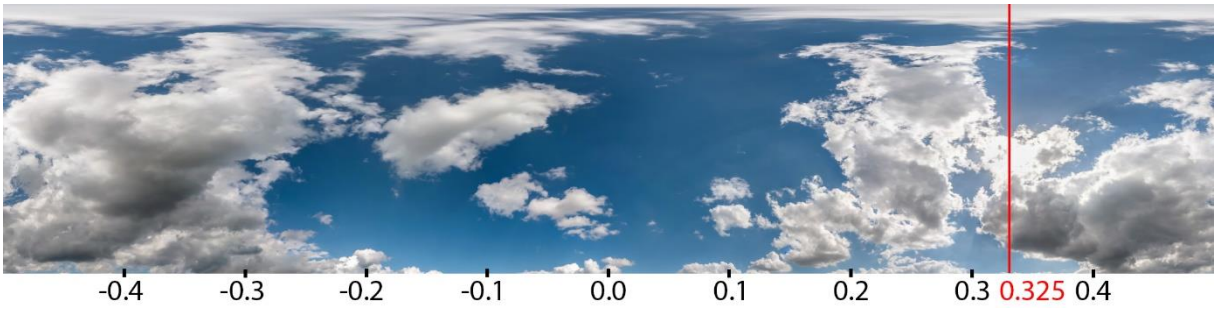
Set the vertical offset

```
texture:setHorizonOffset(-0.0175)
```

Define a sun position

```
texture:setSunPosition(0.325)
```

If a sun position is defined, the texture's horizontal offset is linked to the sun's azimuth.



Adapt the skyshader to the texture's look

```
texture:setSkyAdaption(hue, saturation, value)
```

If you have transparent parts in your texture, the native skyshader is visible. Use this function to adapt its look for that texture.

Set an individual brightness for the texture

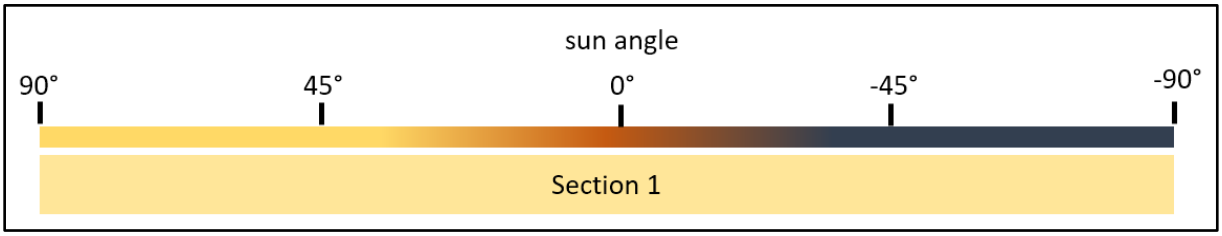
```
texture:setBrightness(0.85)
```

This is a multiplier. If a section uses a custom brightness too, both will be multiplied!

Weather and its sections

A weather consists of one or more sections. In a section one texture is defined to be shown.

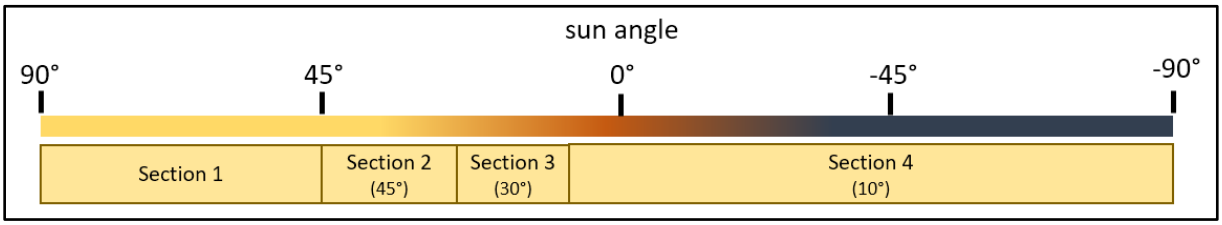
A weather with only 1 section. The bound texture will be used for the complete day/night cycle.




```
section = PURE__2DCLOUDS_set_weather_texture("Windy", "MediumClouds_Noon")
```

This will show the texture with the name "MediumClouds_Noon" (which was prior defined from the file “cloudy”), for the whole day/night cycle, if “Windy”-weather is set.

A weather with multiple sections. The textures are shown, if the current sun angle is lower than the section’s defined sun angle.




```
section = PURE__2DCLOUDS_set_weather_texture("Windy", "MediumClouds_Noon")
section = PURE__2DCLOUDS_set_weather_texture("Windy", "MediumClouds_Mid", 45)
section = PURE__2DCLOUDS_set_weather_texture("Windy", "MediumClouds_Low", 30)
section = PURE__2DCLOUDS_set_weather_texture("Windy", "MediumClouds_Sunset", 10)
```

This will show 4 textures for Windy weather, dependent on the sun angle.

Weather names

Use this weather name strings, to define weather textures:

```
"LightThunderstorm", "Thunderstorm", "HeavyThunderstorm",
"LightDrizzle", "Drizzle", "HeavyDrizzle",
"LightRain", "Rain", "HeavyRain",
"LightSnow", "Snow", "HeavySnow",
"LightSleet", "Sleet", "HeavySleet",
"Clear","FewClouds", "ScatteredClouds",
"Windy", "BrokenClouds", "OvercastClouds",
"Fog", "Mist", "Smoke", "Haze", "Sand", "Dust",
"Squalls", "Tornado", "Hurricane",
"Hail"
```

Section functions

```
section = PURE__2DCLOUDS_set_weather_texture("Windy", "MediumClouds_Noon")
section:setSunPosition(0.05)
section:setBrightness(1.0)
section:setSkyAdaption(0, 0.5, 1.0)
section:setFog(0.25, 0.0, 1.0)
section:setOpacityExponent(1.00)
section:setLightLUT(sunset_light_LUT)
```

Define a section unique sun position

```
section:setSunPosition(0.325)
```

This will overwrite the texture's sun position.

Set an individual brightness for the texture

```
section:setBrightness(0.85)
```

This is a multiplier. If a texture uses a custom brightness too, both will be multiplied!

Adapt the skyshader to the texture's look

```
section:setSkyAdaption(hue, saturation, value)
```

If you have transparent parts in your texture, the native skyshader is visible. Use this function to adapt its look for that texture. This will overwrite the texture's sky adaption.

Set a custom fog look

The fog look is automatically set by Pure, but you can overwrite it individually.

```
section:setFog(horizon_amount, sky_amount, exponent)
```

If you have transparent parts in your texture, the native skyshader is visible. Use this function to adapt its look for that texture. This will overwrite the texture's sky adaption.

Functions Overview

Set functions

PURE__2DCLOUDS_add_texture(name: string, file: string)
add a texture to the set
return: **texture** object

PURE__2DCLOUDS_set_weather_texture(weather_name: string, texture_name: string, sunangle:float)
bind a texture to a weather
return: **section** object

PURE__2DCLOUDS_copy_section_to_weather(weather_name: string, **section**: section_object)
clone an existing section and add it to the weather
return: **section** object

PURE__2DCLOUDS_duplicate_weather(weather_name_from: string, weather_name_to: string)
clone an existing weather
return: **weather** object

PURE__2DCLOUDS_get_weather (weather_name: string)
get the weather object of an existing weather
return: **weather** object

PURE__2DCLOUDS_set_weather_3dcloud_shadows(weather: string, amount: float)
set the 3d cloud shadow amount of a certain weather

PURE__2DCLOUDS_set_default_Sunlight_LUT(lut: table)
set the default sunlight LUT

PURE__2DCLOUDS_set_default_Moonlight_LUT(lut: table)
set the default moonlight LUT

PURE__2DCLOUDS_set_default_sky_adaption(hue: float, saturation: float, value: float)
set the default sky adaption

PURE__2DCLOUDS_setKey(key: string)
set the encryption key

PURE__2DCLOUDS_encryptTextures(key: string, *tex_list: table*)
encrypt textures

Texture functions

texture:**setHorizonOffset**(y: float)
adjust the vertical offset

texture:**setSunPosition**(x: float)
adjust the horizontal offset and link it to the sun angle

texture:**setBrightness**(brightness: float)
adjust the textures brightness

texture:**setSkyAdaption**(hue: float, saturation: float, value: float)
adjust the sky shader

Weather functions

weather:**setBrightness**(brightness: float)
set a multiplier, exclusively for that weather, to change the texture's brightness

weather:**setSkyAdaption**(hue: float, saturation: float, value: float)
adjust the sky shader for all of the weather's sections

weather:**setFog**(horizon: float, sky: float, exponent: float)
adjust the fog look of the weather

weather:**setLightLUT** (lut: table)
set the light LUT for all of the weather's sections

weather:**getSectionAtSunangle** (sunangle: float)
get a weather's section by sun angle
return: **section** object

Section functions

section:**setSunPosition**(x: float)
adjust the horizontal offset and link it to the sun angle

section:**setBrightness**(brightness: float)
set a multiplier, exclusively for that section, to change the texture's brightness

section:**setSkyAdaption**(hue: float, saturation: float, value: float)
adjust the sky shader

section:**setFog**(horizon: float, sky: float, exponent: float)
adjust the fog look of the section

section:**setLightLUT** (lut: table)
set the light LUT

Encryption

You can encrypt your textures, to prevent a further usage. This is important, if you have to manage copyright issues.

Encrypt all textures

```
PURE__2DCLOUDS_encryptTextures("Test123")
```

This will encrypt all textures in the folder:
….\assettocorsa\extension\config-ext\Pure\Skydome_sets**yourset**\textures\

The encrypted files will be placed in the folder:
….\assettocorsa\extension\config-ext\Pure\Skydome_sets**yourset**\textures\encrypted\

!!! Encryption takes its time !!!

While CSP is encrypting, AC will freeze. **Don’t panic!**

Encrypt only specific textures

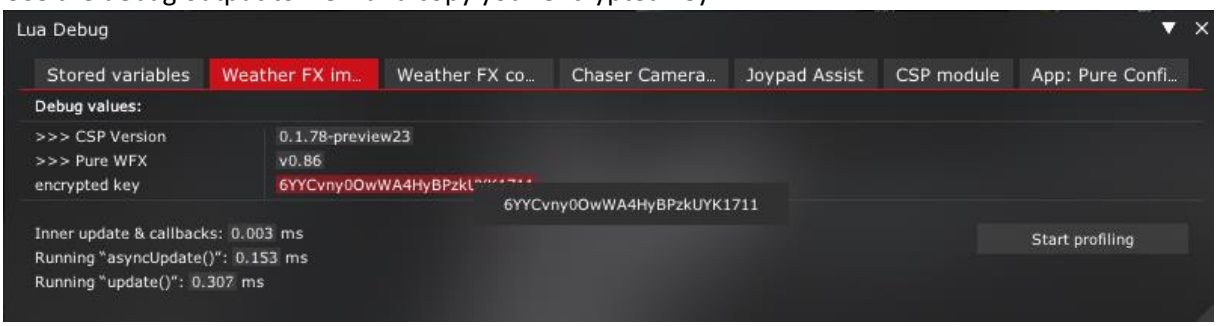
```
PURE__2DCLOUDS_encryptTextures("Test123", {"midcloudy", "clear", "broken"})
```

This will only encrypt midcloudy.dds, clear.dds and broken.dds.

Generate the encrypted key

```
ac.debug("encrypted key", web.encryptKey("Test123"))
```

Use the debug output to view and copy your encrypted key.



Click on the string to copy it to clipboard.

Set the key in your skydome set

```
PURE__2DCLOUDS_setKey("6YYCvny0OwWA4HyBPzkUYK1711")
```

If you use encryption, all textures must be placed in:
….\assettocorsa\extension\config-ext\Pure\Skydome_sets**yourset**\textures\encrypted\

If you use encryption, **ALL** textures must be encrypted!